Bridging Upper Ontology and Modular Ontology Modeling: A Tool and Evaluation

Abhilekha Dalal ✉, Cogan Shimizu, and Pascal Hitzler

Data Semantics Laboratory, Kansas State University, USA
{adalal,coganmshimizu,hitzler}@ksu.edu

Abstract. Ontologies are increasingly used as schema for knowledge graphs in many application areas. As such, there are a variety of different approaches for their development. In this paper, we describe and evaluate UAO (for Upper Ontology Alignment Tool), which is an extension to CoModIDE, a graphical Protégé plugin for modular ontology modeling. UAO enables ontology engineers to combine modular ontology modeling with a more traditional ontology modeling approach based on upper ontologies. We posit – and our evaluation supports this claim – that the tool does indeed makes it easier to combine both approaches. Thus, UAO enables a best-of-both-worlds approach. The evaluation consists of a user study, and the results show that performing typical manual alignment modeling tasks is relatively easier with UAO than doing it with Protégé alone, in terms of the time required to complete the task and improving the correctness of the output. Additionally, our test subjects provided significantly higher ratings on the System Utilization Scale for UOA.

1 Introduction

In many application areas, ontology modeling has become a primary approach to schema generation for data integration and knowledge graphs [11,14,32]. Lately, the policies of Findability, Accessibility, Interoperability, and Reusability (FAIR) have been formulated as essential goals that data receptacles should meet to enhance their data holdings' usefulness [33]. The quest for efficient approaches to model useful and reusable ontologies has, over the years, led to different proposals for ontology creation processes and tooling.

One classic approach is based on so-called upper or foundational ontologies [1,21,31]. Central to this paradigm is the utilizing of ontologies that are generic and large, and as such cover a wide swath of domains, such as BFO [1], DOLCE [10], SUMO [18]. In this approach to modeling, a new (domain) ontology is created in accordance with the mindset or structure conveyed by these upper or foundational ontologies. Technically, alignment of the domain ontology classes and relations to the upper/foundational ontology entities – meaning creating appropriate sub-class and sub-property relationships so that relevant structure or axioms are inherited – play a prominent role.

A more recent approach to ontology modeling is based on a different mindset; modular ontology modeling [27] is based on the idea that an ontology may best be viewed as a collection of interconnected *modules*, each of which correspond to a key notion according to the terminology used by a domain expert. The approach is related to other recent proposals to approach ontology modeling in a *divide and conquer* fashion [22,30] and is a refinement of the eXtreme Ontology Design methodology [24] based on Ontology Design Patterns [15]. In its original conception, and the corresponding tooling, in particular the CoModIDE Protégé plugin [27], the approach de-emphasizes sub-class and sub-property relationships when reusing patterns, and in particular does not account for upper or foundational ontologies.

Different ontology modeling paradigms have different emphases and as such the resulting ontologies have different strengths and weaknesses. Approaches based on foundational ontologies lead to ontologies that are based on a singular philosophical paradigm to ontology building, and thus are internally coherent and deeply thought-out. On the flip side, they are large and monolithic, with little immediately discernible internal structure, and modeling choices are sometimes hard to understand for those who are not ontology engineering specialists. The modular approach, on the other hand, puts less emphasis on overall philosophical coherence, but results in a highly structured ontology that natively aims to reflect conceptualizations by domain experts. Which approach is chosen may also sometimes be subjective, based on perceived advantages or disadvantages, or on particulars of the use case.

In this paper, we provide a case in point that the two just mentioned approaches to ontology engineering are in fact not mutually exclusive, but that it is possible to

use a combination of modular and upper ontology modeling. Combining the best of both worlds can help ensure consistent development of ontologies across multiple domains, and to accomodate a team with differing preferences and perspectives. It will increase the flexibility of training; it will allow more effective governance and quality assurance of ontology development, and it will promote the degree to which multiple different groups of ontology developers and users can inspect and critique.

We thus extend CoModIDE, a graphical paradigm based on Protégé for modeling modular ontologies, with additional functionality, namely the Upper Alignment Tool (UOA) that supports the manual alignment of the currently modeled ontology to a chosen upper ontology, and thus makes it possible to follow both or either of the approaches, as desired.

As CoModIDE did previously not have such alignment capabilities, a user would have needed to use the default Protégé experience to load, identify, and align classes and properties. Thus, we hypothesize (and substantiate in our evaluation) that manual upper ontology alignment with any modular ontology using the plugin developed is comparatively easier than doing it with Protégé alone.

The rest of this paper is organized as follows. Section 2 introduces the UOA. Section 3 discusses relevant work on graphic modeling and ontology development methods and tools. Sections 4 and 5 present our study design for evaluating the tool and the results of our experiment. Section 6 discusses these findings and their implications. Finally, Section 7 sums up the paper and proposes some possibilities for future research. A preliminary demonstration of the Upper Ontology Alignment tool, without evaluation, was already provided in [8].

2 UOA: The Upper Ontology Alignment Tool

Motivation The Upper Ontology Alignment plugin is intended to simplify ontology development for users who want to combine a modular development approach with an upper ontology based one. The UOA thus provides a straightforward interface that gives the user the ability to manually align parts of the currently modeled ontology to a chosen upper ontology. The tool is based on manual alignment, because this is how it is usually done and discussed during modeling with upper ontologies. In principle, algorithms for mapping recommendations could be added, but this is not part of the current functionality.

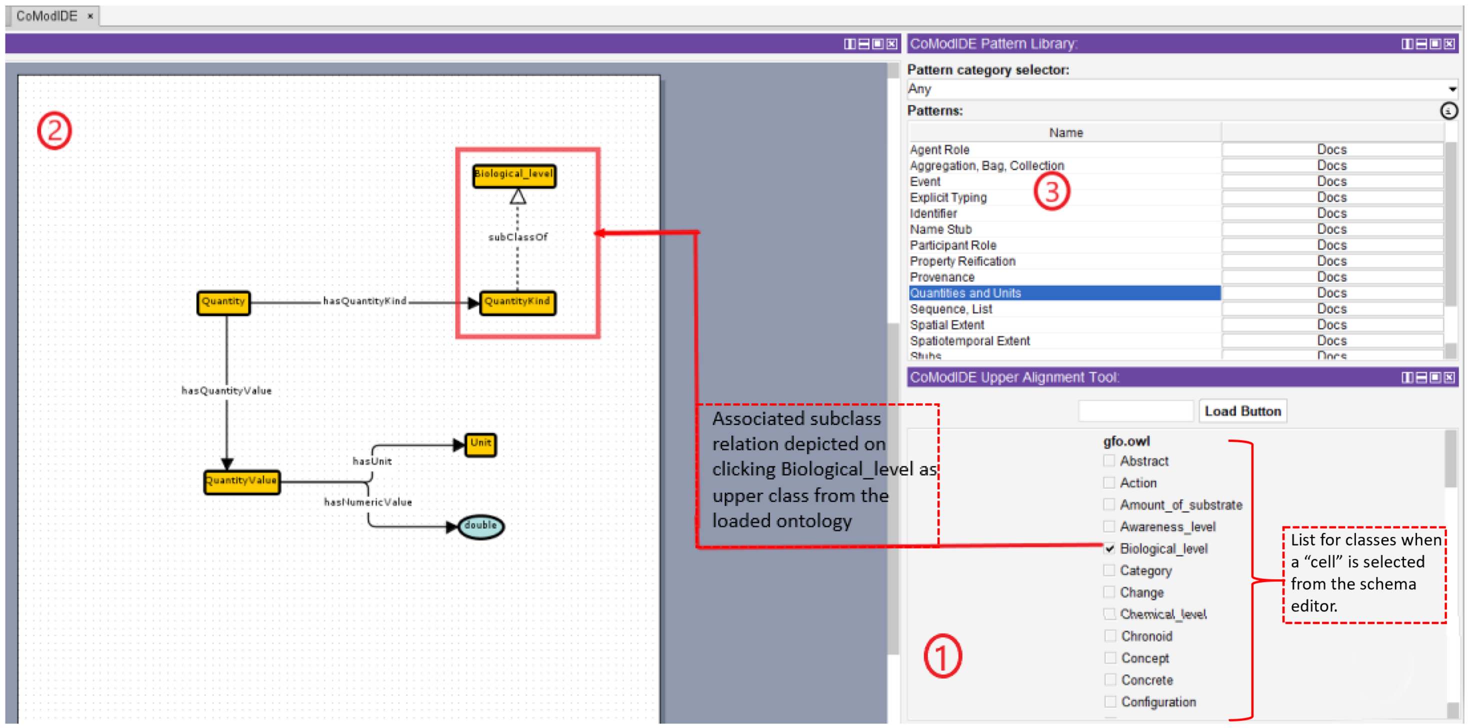
The UOA is provided as part of CoModIDE [27], which is a versatile and established Protégé plugin that supports intuitive and agile visual modeling, reusing ODPs as templates to create modules but does not account for alignment with upper or foundational ontologies [27]. Therefore, we used the following as our design criteria:

1. full integration with Protégé and CoModIDE along with leveraging the graphical user interface of CoModIDE and the creation of pattern-based modules,
2. easy loading of any ontology as an upper ontology and extraction of its concepts and relations, and
3. simple selection of checkboxes to define subClass and subproperty relationships between the currently modeled ontology and the loaded upper ontology.

Implementation The Upper Ontology Alignment (UOA) tool extends CoModIDE, which provides three views: schema editor, pattern library, and the configuration view. UOA is an additional, fourth view, labeled as (1) in Figure 1. Classes are diagrammatically represented as cells, and properties are represented as edges between them. The highlighted red box in the schema editor shows a diagrammatic rendering of an alignment with the loaded upper ontology, UOA user interface displays the list of classes when a cell is selected from the schema editor and likewise, it will switch to list of properties from upper ontology when edges are selected.

Since UOA provides additional functionality to CoModIDE, one of the leading design criteria is that it must be compatible and support the graphical representation of an ontology created with CoModIDE, and should be consistent across all reboots, instruments, and operating systems or versions of Protégé.

The UOA view allows a user to load an ontology file using a load button – which may be an upper or foundational ontology – directly into the view (which is kept isolated from the ontology active in Protégé). The view extracts all of the classes and properties (excluding annotation properties) from the loaded ontology. The user then selects classes (cells) or object/data properties (edges) on the



Name	
Agent Role	Docs
Aggregation, Bag, Collection	Docs
Event	Docs
Explicit Typing	Docs
Identifier	Docs
Name Stub	Docs
Participant Role	Docs
Property Reification	Docs
Provenance	Docs
Quantities and Units	Docs
Sequence, List	Docs
Spatial Extent	Docs
Spatiotemporal Extent	Docs
Stubs	Docs

gfo.owl
- [ ] Abstract
- [ ] Action
- [ ] Amount_of_substrate
- [ ] Awareness_level
- [x] Biological_level
- [ ] Category
- [ ] Change
- [ ] Chemical_level
- [ ] Chronoid
- [ ] Concept
- [ ] Concrete
- [ ] Configuration

Associated subclass relation depicted on clicking Biological_level as upper class from the loaded ontology

List for classes when a “cell” is selected from the schema editor.

Fig. 1: CoModIDE plugin views – 1) UOA, 2) schema editor, 3) pattern library.

graphical canvas. The UOA tool then displays the pertinent entities depending on which glyph is selected on the graphical canvas. The view will automatically construct and add – or remove – the pertinent subClass or subproperty axioms to the ontology when the user selects or deselects checkboxes next to these entities. CoModIDE detects these additions and will also diagrammatically display the added relationships.

The view also provides some supporting functionality for ease and clarity of use: the view will display the currently selected entity, automatically select checkboxes for axioms that are already present in the ontology (e.g., if some entity is already a subClass of the Perdurant class, that particular checkbox will be selected), will display the currently loaded ontology file name, allows for different ontologies to be loaded (i.e., a user is not limited to a single upper ontology), and provides descriptive logging in the case of failure.

3 Related Work

The intention behind having a tool like UOA that supports the interactive visual alignment of ontologies within the modular ontology methodology is to enhance ontology engineers' experience by merging modular ontology modeling with foundational ontologies based modeling. We briefly discuss some other tools that support the same or related goals.

eXtreme Design (XD) [4] was initially proposed to emphasize waterfall methodologies in ontological engineering to introduce a new, more flexible thinking in ontological engineering. It was originally inspired by software engineering methods such as eXtreme Programming (XP) [29] and the experience factory approaches [2]. With the growth of the ontology, instead of a one-time process, an emphasis is placed on the iterative delivery approach to success. The methodology can be divided into three parts. (1) a project initiation and definition phase that is executed only once at the beginning of the project. It is about collecting realistic requirements based on stories that come directly from customers. It is equally necessary to involve domain experts as customers in the development process to confirm the correctness of domain functionality and coverage and the adequacy of terminology and other non-functional requirements.

(2) XD emphasizes the divide and conquer paradigm, takes requirements piece by piece to create modules for each requirement by reusing ODP building blocks, adapting and integrating into the ontology module under development, and like this, all requirements are covered through a development loop that iteratively produces new modules. The module is tested against the selected requirements to ensure that it covers them adequately.

(3) The methodology provides a tangible result in the initial phase and then extends this result with each iteration. As soon as all requirements have been met, the module is released and integrated into the overall solution. XD has been classified as a requirement-driven, inherently modular methodology which focuses on creating reusable modules and reduces failures in ontologies [5,3].

However, the findings also indicate that pitfalls are associated with the possibility of over-reliance on ODPs, as discussed in [12].

Ontology Design Patterns Gangemi [9], and Blomqvist and Sandkuhl [6] introduced Ontology Design Patterns (ODPs) in 2005 to simplify ontology development. ODPs are designed to guide unskilled users by consolidating best practices into reusable building blocks that these users accommodate and specialize in individual ontology development projects. Presutti et al. [25] define a typology of ODPs, including reasoning patterns, naming, transformation, etc. The eXtreme Design methodology [4] describes how ontological engineering projects can be broken down into discrete sub-tasks to be solved using ODPs. Previous studies have shown that using ODPs can reduce the number of modeling errors and inconsistencies in ontologies and that they are found to be useful and helpful by users [3,5].

CoModIDE [26] has been developed as a plugin for the versatile and conventional Protégé environment. The plugin presents three Protégé views and a tab that stores these views. The Schema Editor view provides a graphical overview of the structure of the ontology, including ontology classes, their subClass relationships, and the object type and data type properties of the ontology that associate these classes with data types. All of these objects can be graphically edited by dragging and dropping. The pattern library view offers a number of integrated ontology design patterns from various projects and from the ODP portal.[1] The user can drag and drop design models from the library to the drawing area in order to display these models as modules in their ontology. In the configuration view, the user can set the behavior of other CoModIDE views and their components.

When a pattern is dragged over to the canvas, the constructs of that pattern are copied into the ontology. In addition, they are annotated using the Ontology Pattern Language OPLa [16] to indicate that they belong to a specific module, and are based on a particular pattern. In this way, origin information of the modules is recorded, and the modules can be controlled (folded, unfolded, deleted, commented) as required.

However, the approach de-emphasizes sub-class and sub-property relationships, and in particular does not account for alignment with upper or foundational ontologies.

Prompt-Viz [23] is a visualization tool for the Protégé Prompt [19] plugin that extends PROMPTDiff [20] with information visualization techniques to provide advanced cognitive support for understanding the differences between versions of ontologies. It provides one single visual representation of ontologies within a treemap layout [28]. This visualization aims to provide users the ability to determine the Location, Impact, Type, and Extent (LITE questions) of the changes that have occurred to the ontology. Histogram bars represent the percentage of descendants classified as unchanged, appended, removed, moved-from, moved-to, and directly edited, respectively. It is divided into four linked frames [17]: (1)

[1] http://ontologydesignpatterns.org/

An expandable horizontal tree layout of the ontology showing the differences; it contains a search tool to locate specific concepts quickly. (2) A treemap layout of the ontology installed in a zoomable user interface; (3) A path window showing the location of currently selected concepts in the ontology within the is-a hierarchy and serving as a navigation aid for the treemap component. The treemap view can be zoomed in to show all of the boundaries for each route's level. (4) A comprehensive list of changes that have happened to the currently chosen concept, for instance, the classification of the change, the change procedure, and the reference frame in the previous and new versions of the ontology.

Prompt-Viz offers a sophisticated visualization, but it loses some intuitive aspects of a graphical visualization (e.g., hierarchical relationships between concepts). Using a single visualization to represent the two ontologies, the properties of the source ontologies lose their clarity, which can be sufficient to merge, but makes is less suitable for alignment.

4 Research Method

We conducted a user study to assess the added value of the UOA. The user study consists of four parts: a questionnaire survey to collect necessary data on the subject (e.g., familiarity with ontologies and corresponding tools), two modeling tasks, and a follow-up questionnaire survey to collect information on the usability of Protégé and UOA. The tasks were designed to imitate a standard ontology modeling process in which a conceptual design is developed and approved through whiteboard prototyping. A developer is then to perform a simple alignment task with an upper ontology.

During each modeling task, participants are asked to create an appropriate and correct OWL file for the proposed tasks. To avoid a learning effect, the two tasks use two different schematic diagrams; one is an instance of a pattern, and the other is a small ontology. The specific order in which the user used which tool first and next was randomized among participants (some used Protégé first for the first activity and others used UOA) to avoid bias differences in the activity's complexity. The precision of the developed OWL files and the time required to complete each task were recorded (the latter being limited to 20 minutes per task). Each step of the study has been explained below.

Introductory Tutorial We provided participants with a brief tutorial on the basics required to understand and perform the required tasks. As such, we did not have to impose any prerequisites for participants. The 10-minutes tutorial established a common basic understanding of the basic concepts of ontology modeling and Protégé, including ontologies, top-level ontologies, classes, properties, domains, ranges, sub-class relations.

Prior-Questionnaire Survey The idea of a prior questionnaire survey was to collect information relating to the participant's prior knowledge and experience with topics related to ontology modeling or alignment of ontologies, to be used as control variables in the evaluation. We also asked the participants if they have a Computer Science Background. The questions are listed in Table 1. We used a 5-point Likert scale[2] for the rating.

We also asked the participants to describe their relationship to the test leader, (like student, colleague,

We had two tasks A and B, with two parts to each task. In task A, participants were asked to develop an ontology to model an Event module and then align the entities or properties with an identified upper-level ontology, specifically GFO [13]. For the first part, participants were asked to perform the modeling task using UOA, and for the second they were asked to do it using Protégé alone. Figure 2 (top) shows the expected result.

Similarly, for task B, participants were to develop an ontology to capture information about dog sales, in particular information about pertinent events and roles, and to later align the model with GFO. For the first part, participants were asked to perform the modeling task using the Upper Ontology Alignment Tool, and for the second, they were asked to do it using Protégé alone. Figure 2 (bottom) shows the expected result.

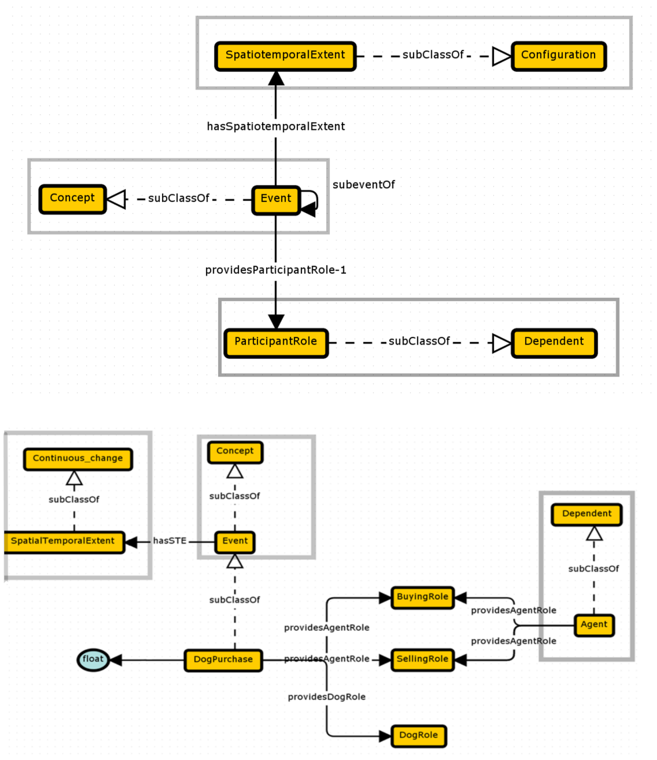


Fig. 2: Tasks A (top) and B (bottom) schema diagrams

Follow-up Questionnaire Survey The follow-up survey included the SUS evaluations for both Protégé and UOA. The SUS is a ubiquitous “quick and dirty” yet reliable tool for measuring a system’s usability. It consists of 10 questions, the responses of which are used to calculate an overall usability score from 0 to 100. The purpose of the selected questions was to capture the tool’s learnability, effectiveness, and efficiency which are the main components of the usability goals.

[2] https://www.simplypsychology.org/likert-scale.html

Table 1: Mean, median, standard deviation and relative standard deviation responses to a priori questionnaire

	mean	median	σ	relative σ
CV1: I have done ontology modelling before	2.14	1	1.46	68%
CV2: I am familiar with Ontology Design Patterns	2.14	2	1.35	63%
CV3: I am familiar with Manchester Syntax	1.52	1	1.03	68%
CV4: I am familiar with Top-level Ontology	1.76	1	1.04	59%
CV5: I am familiar with Protégé	2.24	1	1.58	71%
CV6: I am familiar with CoModIDE pattern library	2.10	1	1.41	67%

Additional information on the SUS and its included items can be found online.[3] Additionally, we inquire about UOA-specific features. These statements are also scored using a Likert scale. However, this data has not been used in our evaluation, except to inform our future work, as described in Section 7. At the end of the survey, participants could provide free comments on UOA's features or their experience with the tool. Our hypothesis underlying the experiment design described above was that UOA improves the approachability of knowledge graph development in such a way that users require less time to produce correct and reasonable output in comparison to using Protégé alone; we also hypothesized that UOA will have a higher SUS score.

5 Results

Participant Distribution The total number of subjects who participated in the user study evaluation was 21, out of which 13 stated that they knew the test leader (the first author); the rest did not report any such relationship. The user study was not limited to subjects having a computer science (CS) background in order to have diversity and capture usability goals across different backgrounds, and we had a small number of participants from fields such as entomology, agriculture engineering, and biochemistry. For self-reported ontological engineering knowledge, the answers are shown in Table 1. The responses differ significantly, with a relative standard deviation (σ/mean) of 59-71%.

Metric Evaluation The metrics that we considered for the result calculation are

1. Time Taken: number of minutes for each modeling task to run were recorded and rounded to the nearest full minute and limited to 20 minutes for a task to run due to practical limitations;
2. Correctness: structural accuracy of the output generated. The complete structurally correct file received 2 points on examining URIs, axioms generated, alignments, 1 point was awarded for a partially correct file (e.g., one or two incorrect linkages, incorrect axiom creation, labels); and 0 points for incorrect files (e.g., absence of axioms, alignment).

[3] https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

Table 2: Summary of statistics comparing Protégé and UOA.

	mean	median	σ
Protégé	17.29	18	4.11
UOA	13.81	15	4.76

(a) Mean, median and standard deviation of *total time-taken* to complete both modeling task.

	mean	median	σ
Protégé (task A)	0.71	1	0.78
Protégé (task B)	0.52	0	0.74
UOA (task A)	1.38	2	0.86
UOA (task B)	1.05	1	0.86

(b) Mean, median and standard deviation of *the output's correctness.*

	CV1	CV2	CV3	CV4	CV5	CV6
T_P	-0.26	-0.07	-0.36	-0.47	-0.23	-0.16
C_P	0.05	-0.02	0.12	0.22	0.05	-0.03
T_U	0.05	0.18	-0.13	-0.15	0.10	0.18
C_U	0.08	-0.03	-0.02	0.03	0.19	0.12

(c) Correlations of control variables (CV) on the Time Taken (T) and Correctness of Output (C) for both Protégé (P) and UOA (U).

	CV1	CV2	CV3	CV4	CV5	CV6
SUS (P)	0.28	0.27	0.28	0.28	0.24	0.26
SUS (U)	0.00	0.02	0.01	0.10	0.13	0.01

(d) Correlations with control variables (CV) on the SUS scores for both tools Protégé (P) and UOA.

	mean	median	σ
Protégé	44.05	42.5	21.04
UOA	71.79	72.5	13.06

(e) Mean, median and standard deviation for SUS score of each tool. The maximum score is 100.

Result	Significance (p)
Time-taken	$p \approx 0.010 < 0.05$
Corr. (Task-A)	$p \approx 0.004 < 0.05$
Corr. (Task-B)	$p \approx 0.012 < 0.05$
SUS Evaluation	$p \approx 0.0000015 < 0.001$

(f) Significance of results.

For the metrics defined, we calculated simple statistics through which data of each modeling task is described. Table 2a and 2b each show the mean, median, and standard deviation of time-taken, and the output's accuracy for each modeling activity through Protégé as well as UOA.

Also, we examined the effects of our control variables (CVs). This analysis is vital because it provides the context for the representation or bias of our dataset. Results can be found in Table 2c, where CV1-CV6 correspond precisely to the questions asked during the prior questionnaire survey (see Table 1). We calculated each CV's bivariate correlation between the sample data and the self-reported data in the survey. We believe calculating correlation has a reasonable measure of impact on the effect, as our sample's limited size is not suitable for partitioning. The partitions (based on the prior questionnaire survey responses) could have been tested in pairs for statistical significance, but the partitions would have been too small to perform the proper statistical tests. However, we emphasize that the sample size strongly influences the correlation effects. SUS scores are analyzed in the same way. Table 2d shows our observed correlations of the SUS score for both tools with our control variables, and Table 2e shows the mean, median, and standard deviation of the data set.

Finally, we compared each metric (time taken and accuracy of output) for one tool against the other, assessing statistical significance; results are given in Table 2f. We see that UOA performs better on both metrics and SUS, with at least $p < 0.05$ in each case, i.e., the results are indeed statistically significant at the 0.05 level. To make the comparison, we calculatee the probability for the null hypothesis that the samples in each dataset come from different underlying distributions, using the paired (two-tail) T-Test, which is a standard tool for this type of analysis and suitable for limited sample size if it is reasonable to assume that values follow a Gaussian distribution, as in our case.

Additional Free-Text Responses Of the 21 participants, 11 decided to leave free-text comments at the end of the questionnaire. We applied qualitative coding and analysis based on the fragments of these comments. That is, we divided the comments based on the line breaks, read the details, and created basic categories. We then allocated the fragments into the categories (with a maximum of one category per segment) [7]. Participants left between 2-5 fragments each to analyze for a total of 35 fragments, 25 of which were encoded, as shown in Table 3.

6 Discussion

Table 3: Free text comment fragments per category

Category	Fragments no.
User-Interface	5
Tree Structure Layout	2
Bugs	5
Additional features	4
Valuable statements	9

Participant Distribution The data show no correlation (bivariate correlation $\leq \pm$ 0.1) between the reported familiarity of the subjects and the reported SUS values; for example, this would have happened if the subjects who knew the author were biased. The high relative standard deviation of knowledge level responses from the prior questionnaire survey shows that our subjects are very diverse in skills. In other words, they are not entirely made up of a limited-experience class or from a particular background of users that UOA will hopefully support at some point. This variation and diversity of education help us evaluate and compare the user's performance and the tool's usability more impartially.

Metric Evaluation To analyze the correlations coefficient between our results and the control variables score collected in the prior survey, we have used threshold values for a correlation $|r|$: 0-0.19 very weak, 0.20-0.39 weak, 0.40-0.59 moderate, 0.60-0.79 strong, 0.80-1.00 very strong.

As depicted in the Table 2c, the metric time-taken used to complete tasks using Protégé correlates negatively with each of the control variables (taken from the prior survey), and the strength of the relationship varies from very weak to moderate. In contrast, the accuracy metric correlates weakly positively as the absolute value ranges from 0 to 0.22 except CV2 and CV6. Analysis for Protégé indicates that familiarity with ontology modeling, top-level ontology, related concepts, and the tool decreases the time required to finish the task and, to any degree, improves the output's accuracy.

However, for the metrics (time-taken and output's correctness) concerning UOA, the relationship's strength and direction are dubious since there are only very weak correlations with control variables varying from 0-0.19. We may interpret that familiarity with ontology modeling does not have much influence and that performance when using UOA is mostly skeptical of the study's control variables. UOA reports having better scores when considering the mean and median for the metric time-taken as described in Table 2a. When examining the underlying data (2f), the significance of the p-value is approx. $0.010 < 0.05$. Subsequent, consider Table 2b for the correctness of both the tasks, UOA performs better for both mean and median score than Protégé. Comparing underlying data for correctness, the statistical significance of the p-value is approx. $0.004 < 0.05$ and $0.012 < 0.05$. Considering both the comparisons, we reject the null hypothesis and confirm that *a user produces correct and reasonable output in less time when using UOA than when using Protégé alone.*

From the above analysis of the correlation coefficient where we observe a very weak correlation between the familiarity of ontology modeling and UOA performance results and the confirmation that the user performs better in terms of time required and output's accuracy when using UOA rather than Protégé, it indicates that UOA has delivered increased accessibility and learnability.

Further, Table 2e illustrates that the SUS scores for UOA have a greater mean, greater median, and smaller σ, attaining a substantial statistical significance of approx. $0.0000015 < 0.001$. Hence, we confirm that the *user finds UOA to have a higher SUS score than when using Protégé alone* from the evaluation. On examining SUS scores (Table 2d), we find that Protégé correlates strongly positively with control variables. The absolute values indicate that subjects do not find the tool very useful in terms of usability goals, including adequate to use, easy to learn, and suitable. In contrast, the SUS correlation coefficient for UOA suggests that there is either a very weak or weak correlation with the CVs. By showing that in less time, users produce accurate output when using UOA and users find UOA to have higher scores, we can say that UOA improves usability and approachability for knowledge graph development, especially for those unfamiliar with ontological modeling.

Additional Free-Text Responses The fragments summarized in Table 3 show the advantages and disadvantages of UOA recognized by subjects as follows:

– *User Interface*: UOA's design format is confusing and less-informative; the button used for loading files into the tool does not serve the purpose much and reduces approachability.
– *Tree Structure Layout*: The users find the view crowded and uncomfortable, not easy to find the classes or properties down in the list.
– *Bugs*: Graphical display on canvas is faulty; checking boxes stopped adding alignments to the model.

– *Additional Features*: There should be a search box to find the classes resp. properties from the list; there should be prompts for the user in case of error; zooming is requested.
– *Valuable Statements*: Users appreciate graphical modeling with the additional feature of alignment with upper ontologies. E.g. *"The Upper Ontology Alignment tool made it much easier to add classes with specific sub-class axiom relations," "This system is very useful and easier to use," "The tool efficiently reduces the manual steps and easy to use. I loved the concept and would highly rate it."*

Some users opted to only leave comments about their performance in the experiment or knowledge about the tool, hence containing no codable fragments. We find that there is an agreement among participants that UOA adds value to graphical modeling and is intuitive and useful. Criticism is aimed at specific, simple bugs or UI functionality.

7 Conclusion and Future Work

Our experiments indicates that UOA allows users to develop ontologies with the option of combining modular ontology modeling with modeling approaches based on upper/foundational ontologies, more correctly and faster than Protégé, irrespective of their previous knowledge level. Our experiments indicates that UOA is more user-friendly and has improved usability goals (SUS score) than the standard Protégé and that UOA concerns affecting users, as opposed to methodological or modeling problems, mainly derive from simple faults in the tool. Overall, this means that modular graphical ontological engineering with alignment to upper ontology using the tool is a practical way to improve ontological engineering accessibility.

Possible extensions of this work, as indicated by feedback from the user study, include: automation or semi-automation of the alignment, provision of more complex alignment capabilities beyond sub-classes or sub-properties, namespace prefix and label presentation, search functionality, and improving the display of the tree structure of classes and properties [8].

Acknowledgement. This work was supported by the U.S. Department of Commerce, National Institute of Standards and Technology, under award number 70NANB19H094.

References

1. R. Arp, B. Smith, and A. D. Spear. *Building ontologies with basic formal ontology.* Mit Press, 2015.
2. V. R. Basili, G. Caldiera, and H. D. Rombach. Experience factory. *Encyclopedia of software engineering*, 2002.
3. E. Blomqvist, A. Gangemi, and V. Presutti. Experiments on pattern-based ontology design. In *Proceedings of the fifth international conference on Knowledge capture*, pages 41–48, 2009.

4. E. Blomqvist, K. Hammar, and V. Presutti. Engineering ontologies with patterns-the extreme design methodology. *Ontology Engineering with Ontology Design Patterns*, (25):23–50, 2016.
5. E. Blomqvist, V. Presutti, E. Daga, and A. Gangemi. Experimenting with extreme design. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 120–134. Springer, 2010.
6. E. Blomqvist and K. Sandkuhl. Patterns in ontology engineering: Classification of ontology patterns. In *ICEIS (3)*, pages 413–416, 2005.
7. P. Burnard. A method of analysing interview transcripts in qualitative research. *Nurse education today*, 11(6):461–466, 1991.
8. A. Dalal, C. Shimizu, and P. Hitzler. Modular ontology modeling meets upper ontologies: The upper ontology alignment tool. In *The 19th International Semantic Web Conference*, volume 2721, pages 119–124, 10/2020 2020.
9. A. Gangemi. Ontology design patterns for semantic web content. In *International semantic web conference*, pages 262–276. Springer, 2005.
10. A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. Sweetening ontologies with DOLCE. In A. Gómez-Pérez and V. R. Benjamins, editors, *Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002, Siguenza, Spain, October 1-4, 2002, Proceedings*, volume 2473 of *Lecture Notes in Computer Science*, pages 166–181. Springer, 2002.
11. C. Gutiérrez and J. F. Sequeda. Knowledge graphs. *Commun. ACM*, 64(3):96–104, 2021.
12. K. Hammar. Ontology design patterns in use: lessons learnt from an ontology engineering case. In *Workshop on Ontology Patterns in conjunction with the 11th International Semantic Web Conference 2012 (ISWC 2012)*, 2012.
13. H. Herre. General Formal Ontology (GFO): A foundational ontology for conceptual modelling. In *Theory and applications of ontology: computer applications*, pages 297–345. Springer, 2010.
14. P. Hitzler. A review of the semantic web field. *Commun. ACM*, 64(2):76–83, 2021.
15. P. Hitzler, A. Gangemi, K. Janowicz, A. Krisnadhi, and V. Presutti, editors. *Ontology Engineering with Ontology Design Patterns – Foundations and Applications*, volume 25 of *Studies on the Semantic Web*. IOS Press, 2016.
16. P. Hitzler, A. Gangemi, K. Janowicz, A. A. Krisnadhi, and V. Presutti. Towards a simple but useful ontology design pattern representation language. In E. Blomqvist, Ó. Corcho, M. Horridge, D. Carral, and R. Hoekstra, editors, *Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017*, volume 2043 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.
17. M. Lanzenberger and J. Sampson. Alviz – a tool for visual ontology alignment. In *Tenth International Conference on Information Visualisation (IV'06)*, pages 430–440. IEEE, 2006.
18. I. Niles and A. Pease. Towards a standard upper ontology. In *2nd International Conference on Formal Ontology in Information Systems, FOIS 2001, Ogunquit, Maine, USA, October 17-19, 2001, Proceedings*, pages 2–9. ACM, 2001.
19. N. F. Noy and M. A. Musen. The PROMPT suite: interactive tools for ontology merging and mapping. *International journal of human-computer studies*, 59(6):983–1024, 2003.
20. N. F. Noy, M. A. Musen, et al. Promptdiff: A fixed-point algorithm for comparing ontology versions. *AAAI/IAAI*, 2002:744–750, 2002.

21. D. Oberle, A. Ankolekar, P. Hitzler, P. Cimiano, M. Sintek, M. Kiesel, B. Mougouie, S. Baumann, S. Vembu, and M. Romanelli. DOLCE ergo SUMO: on foundational and domain models in the SmartWeb Integrated Ontology (SWIntO). *J. Web Semant.*, 5(3):156–174, 2007.
22. D. Osumi-Sutherland, M. Courtot, J. P. Balhoff, and C. J. Mungall. Dead simple OWL design patterns. *J. Biomedical Semantics*, 8(1):18:1–18:7, 2017.
23. D. S. J. Perrin. Prompt-viz: Ontology version comparison visualizations with treemaps. 2004.
24. V. Presutti, E. Daga, A. Gangemi, and E. Blomqvist. eXtreme Design with content ontology design patterns. In E. Blomqvist, K. Sandkuhl, F. Scharffe, and V. Svátek, editors, *Proceedings of the Workshop on Ontology Patterns (WOP 2009) , collocated with the 8th International Semantic Web Conference (ISWC-2009), Washington D.C., USA, 25 October, 2009.*, volume 516 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.
25. V. Presutti, A. Gangemi, S. David, G. A. de Cea, M. Surez-Figueroa, E. Montiel-Ponsoda, and M. Poveda. NeOn Deliverable D2.5.1. a library of ontology design patterns: reusable solutions for collaborative design of networked ontologies. *NeOn Project. http://www. neon-project. org*, 2008.
26. C. Shimizu and K. Hammar. Comodide – the Comprehensive Modular Ontology Engineering IDE. In *ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th International Semantic Web Conference (ISWC 2019) Auckland, New Zealand, October 26-30, 2019.*, volume 2456, pages 249–252. CEUR-WS, 2019.
27. C. Shimizu, K. Hammar, and P. Hitzler. Modular graphical ontology engineering evaluated. In A. Harth, S. Kirrane, A. N. Ngomo, H. Paulheim, A. Rula, A. L. Gentile, P. Haase, and M. Cochez, editors, *The Semantic Web – 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings*, volume 12123 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2020.
28. B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on graphics (TOG)*, 11(1):92–99, 1992.
29. J. Shore et al. *The Art of Agile Development: Pragmatic guide to agile software development*. O'Reilly Media, Inc., 2007.
30. M. G. Skjæveland, D. P. Lupp, L. H. Karlsen, and H. Forssell. Practical ontology pattern instantiation, discovery, and maintenance with reasonable ontology templates. In D. Vrandecic, K. Bontcheva, M. C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L. Kaffee, and E. Simperl, editors, *The Semantic Web – ISWC 2018 – 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I*, volume 11136 of *Lecture Notes in Computer Science*, pages 477–494. Springer, 2018.
31. B. Smith. Classifying processes: an essay in applied ontology. *Ratio*, 25(4):463–488, 2012.
32. R. Vita, J. A. Overton, C. J. Mungall, A. Sette, and B. Peters. FAIR principles and the IEDB: short-term improvements and a long-term vision of OBO-Foundry mediated machine-actionable interoperability. *Database*, 2018:bax105, 2018.
33. M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al. The FAIR guiding principles for scientific data management and stewardship. *Scientific data*, 3, 2016.